Using soccer database

1. Get a list of players names and shirt numbers that play on the team with id = 1

   ```
   SELECT firstName, lastName, shirtNumber
   FROM PLAYER
   WHERE teamId = 1;
   ```

2. Get a list of players names and shirt numbers that play on the team named “Arsenal"

   ```
   SELECT PLAYER.firstName, PLAYER.lastName, PLAYER.shirtNumber
   FROM PLAYER
   JOIN TEAM ON PLAYER.teamId = TEAM.id
   WHERE TEAM.name = 'Arsenal';
   ```

3. Find the team name with the most players

   ```
   SELECT TEAM.name, COUNT(PLAYER.id) AS player_count
   FROM PLAYER
   JOIN TEAM ON PLAYER.teamId = TEAM.id
   GROUP BY TEAM.name
   ORDER BY player_count DESC
   LIMIT 1;
   ```

4. Order the matches that have occurred (happened before today date) in chronological order

   ```
   SELECT *
   FROM MATCH
   WHERE matchDate < CURDATE()
   ORDER BY matchDate ASC;
   ```

5. Get last and first names of the main referee and the match date of each match

   ```
   SELECT REFEREE.lastName, REFEREE.firstName, MATCH.matchDate
   FROM MATCH
   JOIN REFEREE ON MATCH.mainRefereeId = REFEREE.id
   ```

6. Repeat 5 but for matches that have not happened

   ```
   SELECT REFEREE.lastName, REFEREE.firstName, MATCH.matchDate
   FROM MATCH
   JOIN REFEREE ON MATCH.mainRefereeId = REFEREE.id
   WHERE MATCH.matchDate >= CURDATE();
   ```

7. Repeat 5 but for matches that have happened

```
SELECT REFEREE.lastName, REFEREE.firstName, MATCH.matchDate
FROM MATCH
JOIN REFEREE ON MATCH.mainRefereeId = REFEREE.id
WHERE MATCH.matchDate < CURDATE();
```

8. Get last, first name of the main ref, home score , away score from games that happened

```
SELECT REFEREE.lastName, REFEREE.firstName, MATCH.homeScore,
MATCH.awayScore
FROM MATCH
JOIN REFEREE ON MATCH.mainRefereeId = REFEREE.id
WHERE MATCH.matchDate < CURDATE();
```

9. Get last, first name of the main ref, home score , away score from games that happened and home team won

```
SELECT REFEREE.lastName, REFEREE.firstName, MATCH.homeScore,
MATCH.awayScore
FROM MATCH
JOIN REFEREE ON MATCH.mainRefereeId = REFEREE.id
WHERE MATCH.matchDate < CURDATE() AND MATCH.homeScore >
MATCH.awayScore;
```

10. Get the team name, home score , away score from games that happened

```
SELECT TEAM.name, MATCH.homeScore, MATCH.awayScore
FROM MATCH
JOIN TEAM ON MATCH.homeTeamId = TEAM.id
WHERE MATCH.matchDate < CURDATE();
```

11. Get team name,home score, away score from games that happened and home team won

```
SELECT TEAM.name, MATCH.homeScore, MATCH.awayScore
FROM MATCH
JOIN TEAM ON MATCH.homeTeamId = TEAM.id
WHERE MATCH.matchDate < CURDATE() AND MATCH.homeScore >
MATCH.awayScore
```

12. Get team names for teams that competed in match id = 5
    SELECT homeTeam.name AS HomeTeam, awayTeam.name AS AwayTeam
    FROM MATCH
    JOIN TEAM AS homeTeam ON MATCH.homeTeamId = homeTeam.id
    JOIN TEAM AS awayTeam ON MATCH.awayTeamId = awayTeam.id

Using the world database

1. Get the country name with the most cities.

   ```
   SELECT Country.Name, COUNT(City.ID) AS city_count
   FROM City
   JOIN Country ON City.CountryCode = Country.Code
   GROUP BY Country.Name
   ORDER BY city_count DESC
   LIMIT 1;
   ```

2. Get the city with the highest population.

   ```
   SELECT Name, Population
   FROM City
   ORDER BY Population DESC
   LIMIT 1;
   ```

3. Get the language spoken in the country of your choosing.

   ```
   SELECT Language
   FROM CountryLanguage
   WHERE CountryCode = 'USA';
   ```

Using either database

1. Which referees have 10 or more years of experience?
   SELECT firstName, lastName, yearsOfExperiance
   FROM REFEREE
   WHERE yearsOfExperiance >= 10;

2. Who on the team “Real Madrid” has a jersey number of 9?

   ```
   SELECT PLAYER.firstName, PLAYER.lastName, PLAYER.shirtNumber
   FROM PLAYER
   JOIN TEAM ON PLAYER.teamId = TEAM.id
   WHERE TEAM.name = 'Real Madrid' AND PLAYER.shirtNumber = 9;
   ```

3. Which countries have more than 1 language?

   ```
   SELECT Country.Name, COUNT(CountryLanguage.Language) AS language_count
   FROM CountryLanguage
   JOIN Country ON Country.Code = CountryLanguage.CountryCode
   WHERE CountryLanguage.IsOfficial = 'T'
   GROUP BY Country.Name
   HAVING language_count > 1;
   ```

Task 3

1. What users/roles do we have?
   Students
   Professors

2. What tables should we have?

   Students
   Professors
   Departments
   Courses
   Lectures
   Lecture_Offerings
   Student_Registrations

3. What columns in each table?

   <u>Students</u>
   student_id (PK)
   name
   registration_number (Unique)
   course_id (FK from Courses)

   <u>Professors</u>
   professor_id (PK)
   name
   department_id (FK from Departments)

Departments
department_id (PK)
department_name

Courses
course_id (PK)
course_name
department_id (FK from Departments)

Lectures
lecture_id (PK)
title
credits
course_id (FK from Courses)
prerequisite_lecture_id (FK from Lectures, NULL)

Lecture_Offerings
offering_id (PK)
lecture_id (FK from Lectures)
professor_id (FK from Professors)
day_of_week
time
room_number
year
semester

Student_Registrations
registration_id (PK)
student_id (FK from Students)
offering_id (FK from Lecture_Offerings)

4. Make sure you identify your primary and foreign keys

   Done

5. Create a database

   Done

6. Create your tables in MYSQL

   CREATE DATABASE university;

   USE university;

```
CREATE TABLE Students (
    student_id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(100),
    registration_number VARCHAR(50) UNIQUE,
    course_id INT,
    FOREIGN KEY (course_id) REFERENCES Courses(course_id)
);

CREATE TABLE Professors (
    professor_id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(100),
    department_id INT,
    FOREIGN KEY (department_id) REFERENCES Departments(department_id)
);

CREATE TABLE Departments (
    department_id INT AUTO_INCREMENT PRIMARY KEY,
    department_name VARCHAR(100)
);

CREATE TABLE Courses (
    course_id INT AUTO_INCREMENT PRIMARY KEY,
    course_name VARCHAR(100),
    department_id INT,
    FOREIGN KEY (department_id) REFERENCES Departments(department_id)
);

CREATE TABLE Lectures (
    lecture_id INT AUTO_INCREMENT PRIMARY KEY,
    title VARCHAR(100),
    credits INT,
    course_id INT,
    prerequisite_lecture_id INT NULL,
    FOREIGN KEY (course_id) REFERENCES Courses(course_id),
    FOREIGN KEY (prerequisite_lecture_id) REFERENCES Lectures(lecture_id)
);

CREATE TABLE Lecture_Offerings (
    offering_id INT AUTO_INCREMENT PRIMARY KEY,
    lecture_id INT,
    professor_id INT,
    day_of_week VARCHAR(10),
    time TIME,
    room_number VARCHAR(10),
```

```
    year INT,
    semester VARCHAR(10),
    FOREIGN KEY (lecture_id) REFERENCES Lectures(lecture_id),
    FOREIGN KEY (professor_id) REFERENCES Professors(professor_id)
);

CREATE TABLE Student_Registrations (
    registration_id INT AUTO_INCREMENT PRIMARY KEY,
    student_id INT,
    offering_id INT,
    FOREIGN KEY (student_id) REFERENCES Students(student_id),
    FOREIGN KEY (offering_id) REFERENCES Lecture_Offerings(offering_id)
);
```

7. Populate each table with 3-5 rows of data

```
INSERT INTO Departments (department_name) VALUES ('Computer Science'), ('English'), ('Engineering');

INSERT INTO Courses (course_name, department_id) VALUES ('Computer Science', 1), ('Mathematics', 2), ('Digital Engineering', 3);

INSERT INTO Professors (name, department_id) VALUES ('Dan Smith', 1), ('Johnny Applespeed', 2), ('Nathan Gopee', 3);

INSERT INTO Lectures (title, credits, course_id) VALUES ('Computer Science 1', 3, 1), ('Discrete Math for Computer Science', 4, 2), ('Digital Logic Fundamentals', 3, 3);

INSERT INTO Lecture_Offerings (lecture_id, professor_id, day_of_week, time, room_number, year, semester)
VALUES (1, 1, 'Monday', '09:30:00', '151', 2024, 'Fall'),
      (2, 2, 'Tuesday', '12:30:00', '251', 2024, 'Fall'),
      (3, 3, 'Thursday', '01:30:00', '351', 2024, 'Fall');

INSERT INTO Students (name, registration_number, course_id) VALUES ('Linus', 'REGIST001', 1), ('Amy', 'REGIST002', 2), ('Lucy', 'REGIST003', 3);

INSERT INTO Student_Registrations (student_id, offering_id) VALUES (1, 1), (2, 2), (3, 3);
```

8. Upload to BitBucket

   Using Github

9. Delete a row of data

   DELETE FROM Students WHERE student_id = 2;

10. Update a row of data

    UPDATE Students SET name = 'Hello World' WHERE student_id = 1;

11. Delete a table

    DROP TABLE Professors;

12. Delete your whole database

    DROP DATABASE university;

13. Reget your database from BitBucket.

    Git clone the repository

14. Practice joining tables and answering questions using your database

    SELECT Professors.name, Lectures.title
    FROM Professors
    JOIN Lecture_Offerings ON Professors.professor_id = Lecture_Offerings.professor_id
    JOIN Lectures ON Lecture_Offerings.lecture_id = Lectures.lecture_id;

    SELECT Students.name, Lectures.title
    FROM Students
    JOIN Student_Registrations ON Students.student_id = Student_Registrations.student_id
    JOIN Lecture_Offerings ON Student_Registrations.offering_id =
    Lecture_Offerings.offering_id
    JOIN Lectures ON Lecture_Offerings.lecture_id = Lectures.lecture_id;